VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Java herní portál

Java game portal

2017 **Michal Rath**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Rath**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Java herní protál
Java Game Portal

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem je vytvořit internetový herní portál umožňující hraní již vytvořených her v jazyce Java.

Systém umožní:
1. Správu uživatelů.
2. Správu dostupných her.
3. Správu odehraných her.
4. Vytváření turnajů (každý s každým, pavouk).
5. Rovněž bude navrženo a použito rozhraní na již existující hry, které bude umožňovat:
a) pouštět hru jako Applet,
b) pouštět hru pomocí technologie Java WebStart,
c) zakládat nové hry,
d) sledovat průběh hry pomocí Java Appletu,
e) získávat průběžné obrázky probíhající hry, bodový stav, pořadí, atd.
6. Připojování hráčů a jejich znovu připojení po ztrátě spojení.
7. Nahrazení hráčů umělou inteligencí po jejich trvalém odpojení (ztrátě spojení), případně znovu napojení hráče, který byl už jednou nahrazen UI a znovu se mu podařilo připojit.
8. Textovou konverzaci hráčů.
9. Naplánování hry na určitý čas.

Práce bude obsahovat:
1. Popis použitých technologií.
2. Implementaci výše popsaného programu.
3. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9. Dostupné z: http://it-ebooks.info/book/2249/

Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 28. 4. 2017

.....
podpis studenta

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Davidu Ježkovi, Ph.D. za odbornou pomoc, ochotu, trpělivost a konzultaci při vytváření této bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá vytvořením herního portálu, který je napsán v jazyce Java. Herní portál je postaven na principu server-klient a její uživatelské rozhraní má desktopové provedení a webové provedení. V první části práce jsou popsány teoretické základy, které jsou potřebné pro tuto aplikaci. Tyto základy obsahují popis objektově orientovaného programování, Java technologií, webových aplikací a databází. V další části práce je popsána samotná praktická implementace portálu a jednotlivé funkce, které portál poskytuje. Součástí jsou také UML diagramy a ukázky kódů.

Klíčová slova

Java, server, herní portál, sockety, JSP.

Abstract

This bachelor thesis is about creation of game portal, which is written in Java programing language. Game portal is built on server-client principle and its user interface has desktop version and web version. In the first part of this thesis are described theoretical basics, which are needed for this application. These basics includes description of object-oriented programming, Java technologies, web applications and databases. In next part of thesis is described practical implementation of portal and its particular functions, which portal provides. This thesis also includes UML diagrams and examples of source codes.

Key words

Java, server, game portal, sockets, JSP.

Seznam použitých zkratek

Zkratka	Anglický význam	Český popis
API	Application Programming Interface	Aplikační programové rozhraní
DAO	Data Access Object	Objekt pro přístup k datům z databáze
GUI	Graphical User Interface	Grafické uživatelské rozhraní
HTML	HyperText Markup Language	Značkovací jazyk pro hypertext
HTTP	Hypertext Transfer Protocol	Protokol pro výměnu HTML dokumentů
IDE	Integrated Development Environment	Vývojové prostředí
JDBC	Java Database Connectivity	API Javy definující přístup k databázi
JDK	Java Development Kit	Celek složený z JVM a API pro vývoj
JDO	Java Data Objects	Rozhraní pro persistenci objektů
JMS	Java Messaging Services	API pro posílání zpráv mezi 2 klienty
JPA	Java Persistence API	Framework pro ORM
JSP	Java Server Pages	Technologie Javy pro tvorbu webových stránek
JSTL	JSP Standard Tag Library	Knihovna značek pro JSP stránky
JVM	Java Virtual Machine	Virtuální stroj Javy
ODBC	Open Database Connectivity	Standardizované API pro přístup k databázi
ORM	Object-relational mapping	Objektově-relační mapování
RMI	Remote Method Invocation	Volání objektů z jiného JVM
SQL	Structured Query Language	Jazyk pro práci s databázemi
UML	Unified Modeling Language	Jazyk pro vizualizaci a modelování
URL	Uniform Resource Locator	Identifikátor umístění zdrojů
XML	Extensible Markup Language	Rozšiřitelný značkovací jazyk

Obsah

1 Úvod

V dnešní době je hraní počítačových her běžnou součástí života většiny lidí, kteří tuto aktivitu vyhledávají za účelem odreagování se od každodenní rutiny a stereotypu svého života. Čím dál tím více roste poptávka po hrách, které jsou volně dostupné, hardwarově nenáročné a ideálně hratelné bez nutnosti instalace. Z těchto podmínek logicky plyne poptávka po hraní webových her přes internetový prohlížeč.

Cílem této bakalářské práce je vytvořit funkční herní portál v programovacím jazyce Java. Tento herní portál je v podstatě rozdělen do čtyř částí: 1 - serverová část, 2 - klientská část, 3 - webová část a 4 - obslužné knihovny. Součástí bakalářské práce je i jedna jednoduchá ukázková hra, na které je demonstrována funkčnost herního portálu.

Po dohodě s vedoucím byly z řešení vynechány body 5 - a), 5 - b) a 5 - d). Nově (od roku 2014) musí být Java applety podepsané a zadání bakalářské práce bylo vytvořeno ještě před tímto ustanovením.

V této práci bude popsán základ programovacího jazyka Java, včetně použitých technologií. Dále budou podrobněji popsány jednotlivé části herního portálu. Serverová část je jádrem celého portálu, v které se vykonávají všechny úkoly za pomoci obslužných knihoven, dále taky slouží k ukládání a načítání z databáze. Klientská část je grafické rozhraní, které umožňuje spouštět hry a také veškerou obsluhu. Webová část je webové rozhraní sloužící k zakládání nových her nebo turnajů a také k registraci či prohlížení hráčského profilu.

2 Teoretický základ

2.1 Programovací jazyk Java

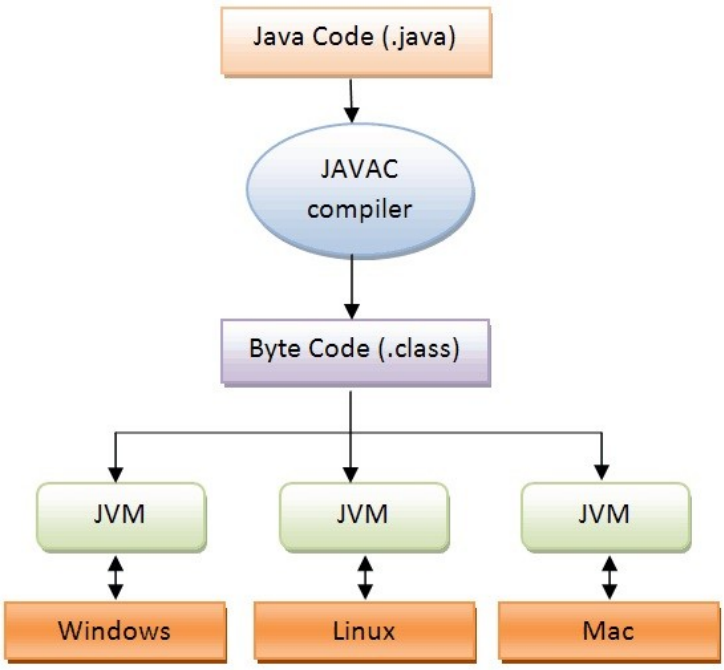
Java je objektově orientovaný programovací jazyk, který vychází z jazyka C++. Java byla oficiálně představena veřejnosti v roce 1995, dříve byla vyvíjena pod názvem Oak a to od roku 1991. V roce 1995 byla však přejmenována na Java a vydána společností Sun Microsystems. Od roku 2007 je Java vyvíjena jako open source a od roku 2010 je vlastníkem Javy společnost Oracle.

Java je v současné době jedním z nejpopulárnějších programovacích jazyků. Jedním z důvodů, proč je Java tak populární, je její nezávislost na zvolené platformě. Java má několik edicí pro různé zařízení např.:

- Java Card - Java pro čipové karty
- Java ME (MicroEdition) - Java pro mobilní zařízení
- Java SE (Standard Edition) - Java pro osobní počítače
- Java EE (EnterpriseEdition) - Java pro podnikatelské aplikace a informační systémy

Java má vlastní platformu složenou ze dvou částí: virtuální stroj - Java VirtualMachine (JVM) a Java ApplicationProgramming Interface (API).API je kolekce softwarových komponent, které mají spoustu užitečných funkcí. Komponenty jsou rozděleny do knihoven podle souvisejících tříd a rozhraní, tyto knihovny se nazývají balíčky (packages). [1]

Virtuální stroj provádí samotný program nezávisle na systému. Java programy jsou napsány obyčejným textem s příponou *.java,* poté jsou zkompilovány do bytecode a uloženy do souboru s příponou *.class*. Bytecode je jazyk virtuálního stroje, který soubory *.class* provádí (viz. obrázek 2.1).[1]



Obrázek 2.1: Kompilace kódu a nezávislost na systému [4]

2.2 Vývojové prostředí

Na trhu existuje velké množství vývojových prostředí a to komerčních i open source. Pro tuto bakalářskou práci jsem používal NetBeans IDE. NetBeans je vývojové prostředí, které je k dispozici zdarma. Pro tvorbu bakalářské práce jsem používal NetBeans ve verzi 8.0 a Javu ve verzi 8.

2.3 Objektově orientované programování

"Objektově orientované programování vychází z myšlenky, že všechny programy, které vytváříme, jsou simulací buď skutečného, nebo nějakého námi vymyšleného virtuálního světa. Čím bude tato simulace přesnější, tím bude výsledný program lepší." [2] Všechny objekty reálného světa se dají přepsat jako instance třídy. Třída je skupina objektů stejného typu např.: strom. Každý objekt je určen svým jménem, vlastnostmi (atributy) a chováním (metody). Nejlépe si můžeme ukázat objektově orientované programování na příkladu: Lípa je objekt třídy strom s atributy věk a výška a metodami např.: zjisti věk.

2.3.1 Zapouzdření

Jednou z vlastností objektově orientovaného programování je zapouzdření, to znamená, že atributy nebo metody daného objektu jsou zabezpečeny proti zásahu zvenčí. Existují tři typy: veřejný (public), soukromý (private) a chráněný (protected). K veřejným metodám a datům se může přistupovat odkudkoliv (z jakékoliv třídy). K soukromým pouze z dané třídy, ve které jsou vytvořeny. K chráněným metodám a datům má přístup třída, ve které jsou vytvořeny a třídy, které z této třídy dědí nebo jsou ve stejném balíčku (package).

2.3.2 Dědičnost

Dědičnost znamená, že třída, která dědí (potomek) si ponechává všechny atributy a metody třídy, ze které dědí (rodič) bez nutnosti je znovu implementovat a navíc je rozšiřuje o vlastní metody a atributy. V Javě na rozdíl od jiných jazyků může třída dědit jen z jedné třídy.

2.3.3 Rozhraní

Rozhraní (interface) je konstrukce, která obsahuje pouze proměnné a hlavičky metod, bez vlastní implementace. Třída, která rozšiřuje rozhraní, potom tyto metody implementuje.

2.3.4 Polymorfismus

"V programovacím jazyce se jedná o možnost volat stejné metody u různých objektů, aniž bychom věděli, jakého přesně jsou typu. Navíc může mít stejná metoda u různých objektů odlišný význam. To je možné díky tomu, že vždy známe společného předka těchto různých objektů. Tím může být třída, abstraktní třída nebo rozhraní."[3]Pokud v Javě metoda překrývá metodu předka, bývá značena anotací `@Override`.

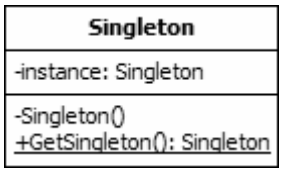
2.3.5 Skládání

Třídy mohou obsahovat kromě běžných datových typů také jiné třídy. Například třída `Strom` může mít jako atributy `int věk`, `string jméno` a také `List list`, což je třída s vlastními atributy.

2.4 Návrhový vzor singleton

Návrhový vzor singleton (česky překládáno jako jedináček) byl definován čtveřicí programátorů, kteří jsou známí pod názvem Gang of Four. Tento návrhový vzor spadá do skupin tzv. tvořících vzorů (anglicky creational). Jeho účelem je zajistit, že daná třída bude vytvářet pouze a jenom jednu instanci, jinými slovy to znamená, že všechny reference na daný objekt v rámci aplikace budou odkazovat na stejnou instanci. Používá se typicky v situaci, kdy máme jedno místo v naší aplikaci nebo jeden objekt, který poskytuje přístup k nějakým zdrojům nebo funkčnosti. [22]

Ve zdrojovém kódu se tento vzor implementuje tak, že u dané třídy definujeme pouze privátní konstruktor, statickou proměnnou, která drží instanci této třídy a statickou metodu `getInstance()`, která vrací tuto instanci. Pokud instance ještě není vytvořena, tak tato metoda zavolá privátní konstruktor. Jak vypadá třída pomocí UML je zobrazeno na obrázku 2.2.



Obrázek 2.2: Návrhový vzor singleton [22]

Ukázka implementace:

```
public classSingleton() {
      private static Singleton instance;
      privateSingleton() {
      }
      public static SingletongetInstance() {
            if (instance == null) {
                  instance = newSingleton();
            }
            return instance;
      }
}
```

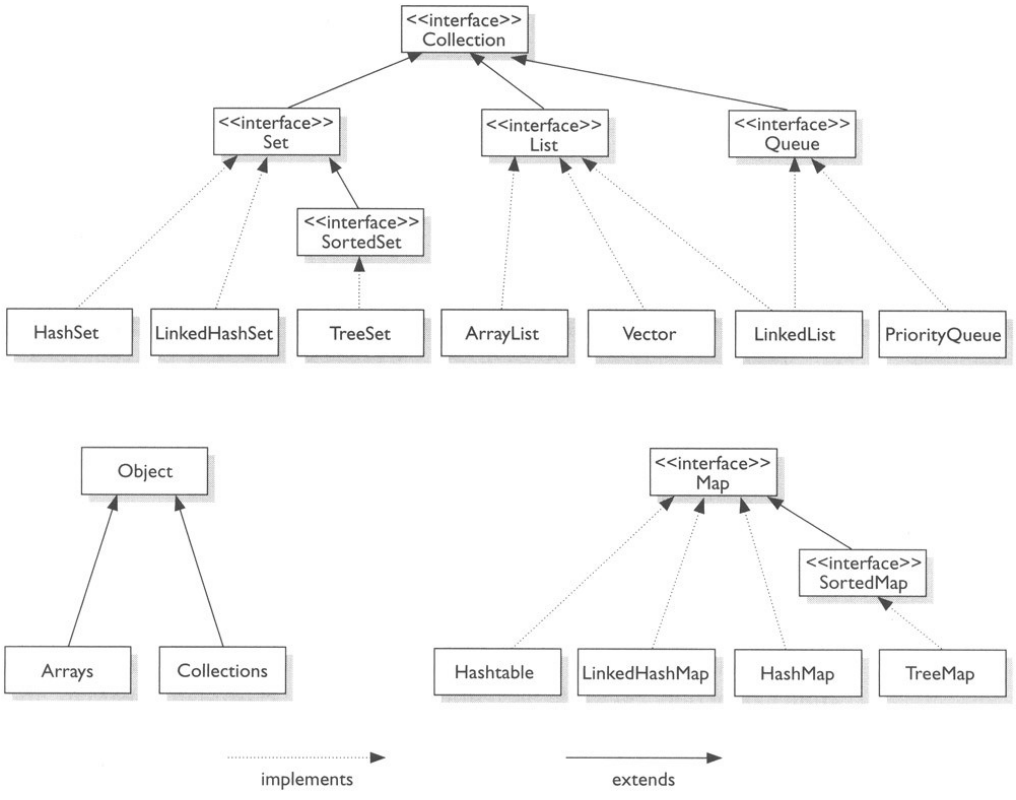
3 Prvky jazyka Java

3.1 Datové typy

Datové typy mohou být buď primitivní, nebo referenční. Mezi primitivní patří boolean, byte, short, int, long, float, double a char. Referenční typy jsou buď objekty, nebo pole. Mezi nejznámější typy reprezentované objektem patří řetězec (String), který je obsažen v balíku `java.lang.String` nebo třeba datum (date) z balíku `java.util.Date`. Svůj název mají, protože se s nimi pracuje pomocí odkazů (referencí).

3.2 Kontejnery

Kontejnery označované též jako kolekce jsou objekty obsažené v API balíku `java.util` a slouží k ukládání jiných objektů. Jejich velikost není, na rozdíl od polí, předem dána, ale dynamicky se zvětšuje za běhu, pokaždé když přidáme objekt do kolekce. Kolekce jsou velmi populární, protože práce s nimi je jednoduchá. K jednotlivým objektům se nemusí přistupovat na základě indexu a většina metod pro práci s kolekcemi je obsažena v API.

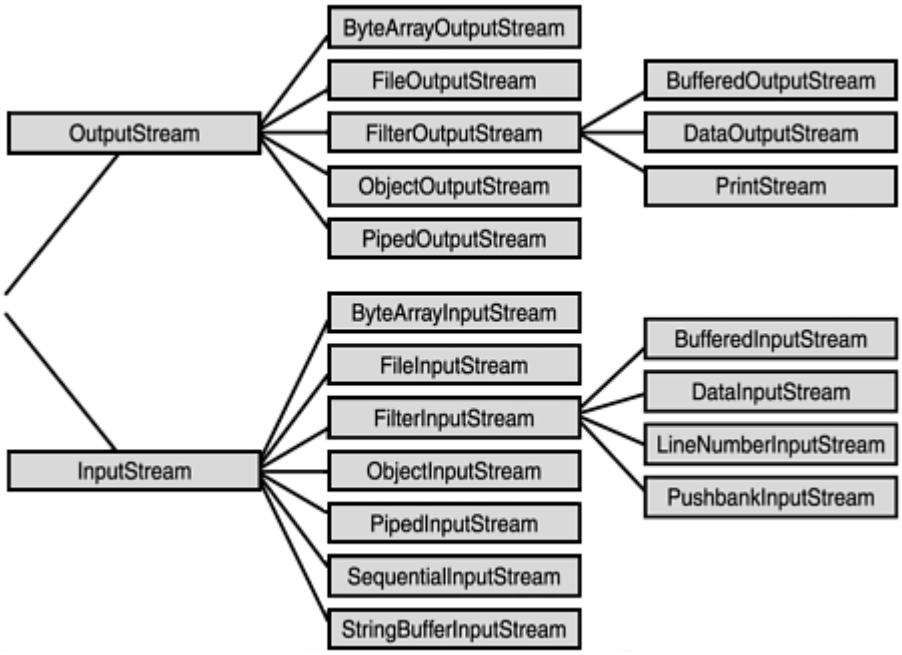


Obrázek 3.1: Kolekce v jazyce Java [5]

Do kolekcí nemůžeme ukládat primitivní datové typy tak jako do polí. Jednotlivé primitivní datové typy musí být uloženy jako instance odpovídající třídy např.: typ `int` musí být převeden na objekt třídy `Integer`, `char` na `Character`, atd.

3.3 Proudy

Pro vstup a výstup dat se v Javě používají proudy (streams). Proudy se dělí na znakové a binární. Znakové posílají data po jednom znaku a jsou vhodné hlavně pro práci s textem. Binární proudy posílají data po jednom bajtu a mohou přenášet libovolná data rozložená na jednotlivé bajty. Proudy se využívají při práci se soubory např.: ukládání dat do databáze anebo třeba při vstupu z klávesnice, či výstupu na obrazovku. [6]



Obrázek 3.2: Binární proudy v jazyce Java [6]

3.4 Sockety

Sockety slouží ke komunikaci mezi serverem a klientem a k posílání dat využívají proudy. Na straně serveru se vytváří `ServerSocket` za pomoci čísla portu, který poslouchá, jestli se nějaký klient připojil. [7]Klient vytvoří `Socket`, který se pokusí připojit na server, pokud se mu to podaří, vytvoří si server u sebe socket a nyní spolu mohou komunikovat, tak že budou zapisovat do a číst z daného socketu. Sockety jsou reprezentovány třídou `java.net.Socket`. [8]

3.5 Vlákna

"Vícevláknový program obsahuje dvě a více částí, které mohou běžet současně. Každá část takového programu se nazývá vlákno (thread) a každé vlákno definuje samostatnou cestu provádění." [9] V Javě jsou vlákna reprezentována třídou `Thread` a rozhraním `Runnable` z balíku `java.lang`. Pokud chceme vlákna používat musí naše třída buď dědit z třídy `Thread`, nebo rozšiřovat rozhraní `Runnable`, v obou případech musíme přepsat metodu `run()`. Java umožňuje dědit pouze z jednoho předka a potřebujeme-li aby naše třída dědila, využijeme radši možnost rozšířit rozhraní. Vlákno spustíme metodou `start()`.

3.6 Výjimky

"Výjimka je chyba, k níž dojde za běhu programu." [9] Výjimky se v Javě dají ošetřit kódem a při jejich vyvolání se vykoná jiný kus kódu. Výjimky jsou reprezentovány jako třídy, které jsou odvozené ze třídy `Throwable`. Základní výjimky jsou dvojího typu: `Exception`, které se dají ošetřit a `Error`, která souvisí s JVM a není v moci programátora ji nějak ošetřit. [9]

Výjimky třídy Exception se dají a musí ošetřit. Můžeme použít buď již předdefinované výjimky, nebo si můžeme vytvořit vlastní. Základní slova při práci s výjimkami jsou: try, catch a finally. V bloku `try` je problematický kód, který může vyhazovat výjimku. V bloku `catch` je odchycená výjimka zpracována, např.: vypiš chybu nebo vrať se do menu. Blok `finally` není povinný a slouží k vykonání kódu, který se má provést vždy bez ohledu na to, zda byla výjimka vyvolána nebo ne, např.: uzavření spojení.

3.7 Časování

Občas nastane situace, kdy potřebujeme spustit metodu v určitý čas nebo opakovaně po určitém čase. K naplánování metody na určitý čas můžeme využít třídu `Timer` z balíku `java.util`. Třída `Timer` může sloužit jak k opakovanému spouštění po určitém čase tak ke spuštění jednou v určitý čas.

Třída `Timer` slouží pouze ke spouštění, ale abychom určili, co chceme v daný čas provést, musíme využít třídu `TimerTask`. `TimerTask` se taktéž nalézá v balíku `java.util`. `TimerTask` je abstraktní třída, která implementuje rozhraní `Runnable`. Pokud chceme vytvořit vlastní `TimerTask`,musíme tuto třídu rozšířit. Náš `TimerTask` může být poté naplánován použitím třídy `Timer`." [10]

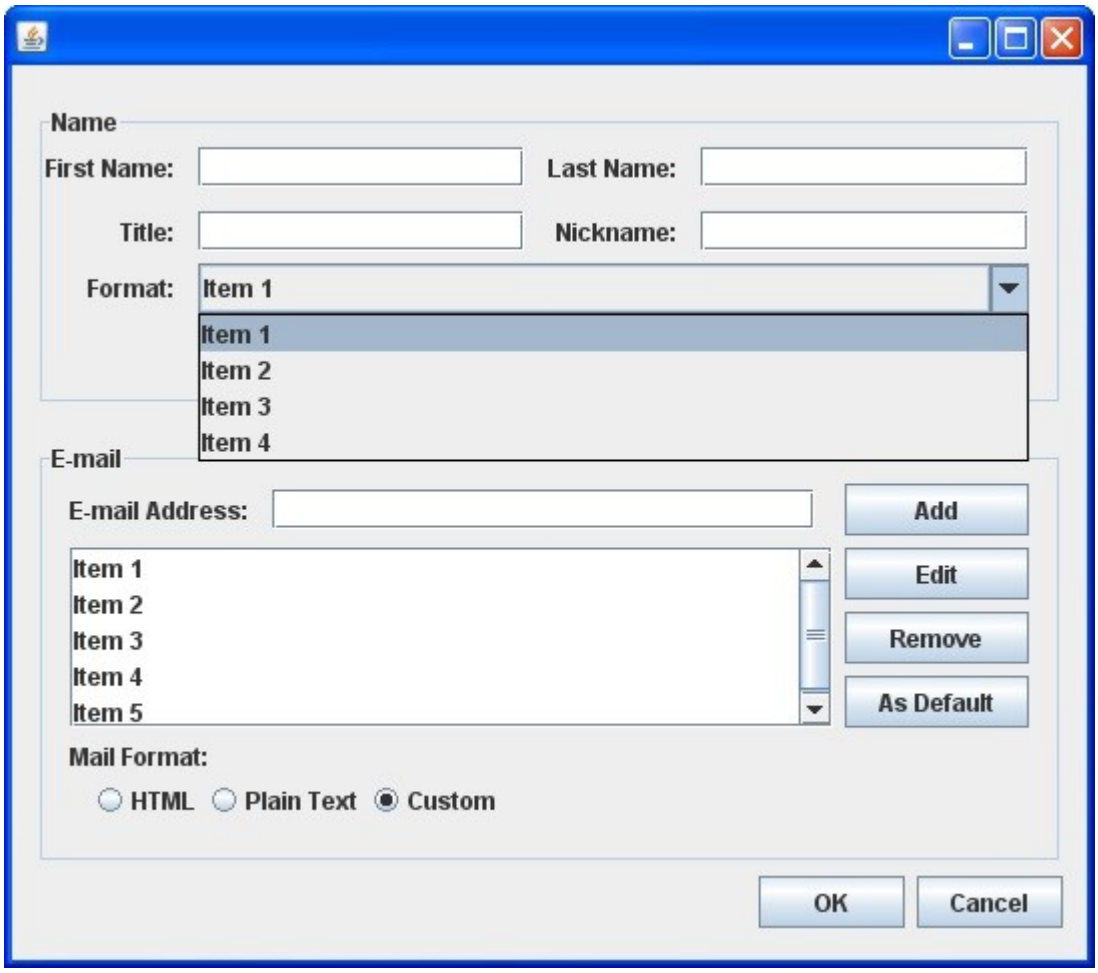
Abychom spustili určitou metodu v daný čas, musíme si napřed vytvořit novou třídu, která bude dědit z třídy `TimerTask`. V této třídě překryjeme metodu `run()` tím, co chceme, aby bylo v daný čas vykonáno. Poté vytvoříme novou instanci této třídy.Pokud chceme vložit nějaké atributy, musí být předány v konstruktoru. Také si vytvoříme novou instanci třídy `Timer` a pak třídě `Timer` pomocí metody `schedule()` předáme instanci naší třídy a čas kdy se má spustit. Čas musí být uložen v datovém typu date.

3.8 Grafické uživatelské rozhraní

"Grafické uživatelské rozhraní (GUI) je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků. Na monitoru počítače jsou zobrazena okna, ve kterých programy zobrazují svůj výstup. Uživatel používá klávesnici, myš a grafické vstupní prvky jako jsou menu, ikony, tlačítka, posuvníky, formuláře a podobně." [26]

Grafické rozhraní v Javě je realizováno pomocí knihovny Swing. Tato knihovna se nachází v balíku `javax.swing`. Všechny zobrazované prvky jsou komponenty. Mezi komponenty patří tlačítka, textové plochy, titulky, atd. Aby byly tyto komponenty správně zobrazeny, musí být přiřazeny do kontejnerů. Mezi takovéto kontejnery můžeme zařadit třeba `JPanel` nebo`JFrame`.

Pro umisťování prvků na plochu slouží třída `BorderLayout`. Jednotlivé komponenty mohou být umístěny na různé pozice v rámci kontejnerů a kontejnery mohou být umístěny na různé pozice v rámci celého okna. Jedná se o rozmístění relativní. Pokud chceme prvky umístit na pevně danou pozici, využijeme metodu `setBounds`. Ve vývojových prostředích existují různé editory, které nám toto rozmisťování na dané pozice ulehčují. Nastavit si pochopitelně můžeme i velikost okna, k tomu také slouží metoda `setBounds`.



Obrázek 3.3: Ukázka grafického rozhraní [27]

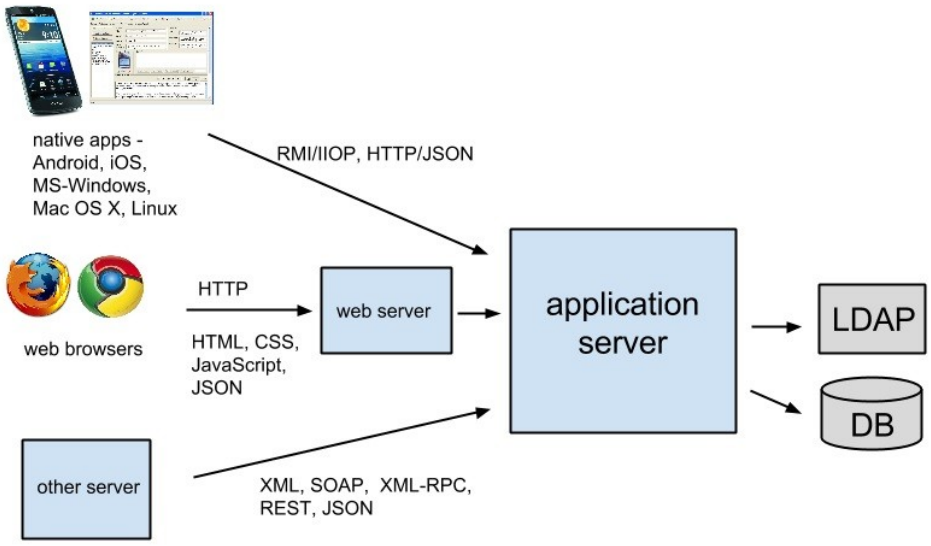
Všechny grafické prvky jsou objekty. Vytvoření tlačítka vlastně znamená vytvoření instance třídy `JButton`. Na tyto objekty jsou volány události, které zajišťují funkčnost programu. Třeba při stisku tlačítka se zavolá událost, která provede nějakou metodu, například sečte dvě čísla. V každé třídě, která se zabývá GUI, by mělo být nadefinováno co se má stát při zavření. Například tímto kusem kódu: `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` říkáme, aby byl program ukončen a ne jen aby bylo zavřeno grafické rozhraní.

4 Webové aplikace

„Webové aplikace jsou aplikace, kde uživatelským rozhraním je webový prohlížeč.“ [11]

Hlavními výhodami webových aplikací je, že koncový uživatel nemusí aplikaci instalovat – potřebuje pouze webový prohlížeč, přes který k aplikaci přistoupí z jakéhokoliv počítače nebo stroje. Výhodou pro vývojáře je to, že se stará pouze o jednu verzi aplikace a při aktualizaci aplikace nemusí řešit doručení a provedení této aktualizace ke koncovému uživateli – pouze aktualizuje verzi běžící na serveru. Další výhodou může být i to, že aplikace je pro koncového uživatele strukturována jako množina webových stránek, které mají své jednoznačné URL adresy, tzn. že na každou stránku se lze dostat zpravidla přímým odkazem. [11]

Z pohledu vývoje jsou aplikace obvykle tvořeny pomocí třívrstvé architektury. Jednotlivými vrstvami, které jsou vidět na obrázku č.4.1, jsou: prezentační vrstva, aplikační (někdy též business) vrstva a datová vrstva. Prezentační vrstva se stará o samotné grafické rozhraní aplikace a u webových aplikací jde o zobrazenou webovou stránku, která je zpravidla napsaná pomocí HTML kódu. Aplikační vrstva má na starosti vykonávání jednotlivých funkcí, které jsou zpravidla volány po nějaké akci, kterou uživatel provedl v prezentační vrstvě. Tyto funkce jsou vykonávány na straně serveru a nijak nezatěžují klienta. Poslední vrstvou je vrstva datová, která se stará o uložení a vrácení dat. Data jsou většinou ukládaná pomocí nějaké relační databáze, ale stejně dobře můžou data být uložena v textovém souboru nebo např. v XML souboru. Datová vrstva přijímá „pokyny“ co s daty vykonat z vyšší – aplikační vrstvy.



Obrázek 4.1: Ukázka třívrstvé architektury [11]

Jednotlivé vrstvy můžou být pouze logicky odděleny v rámci jedné aplikace/jednoho projektu a můžou všechny běžet na jednom stroji/serveru, ale častým jevem je, že jsou vrstvy odděleny

i na fyzické vrstvě a zpravidla například datová vrstva (databáze) běží na jiném serveru než aplikační vrstva.

4.1 Webové aplikace v jazyce Java

Programovací jazyk Java je použitelný s drobnými modifikacemi pro několik různých platforem, což je současně i jeho výhodou. Java tedy umožňuje vytvářet i webové aplikace a také pro ně má poměrně velkou podporu knihoven a funkcí.

Pro samotný běh webové aplikace potřebujeme odpovídající aplikační server, který bude umět provádět Java bytecode. Tyto aplikační servery jsou zpravidla freeware nebo open source, ale existují i komerční verze. Asi nejoblíbenějším Java aplikačním serverem je Tomcat. Dalšími servery jsou pak např. JBoss, GlassFish, WebLogic, IBM WebSphere a další.

Samozřejmostí pro běh webové aplikace a vůbec aplikačního serveru je nainstalovaný virtuální stroj Javy.

4.1.1 Servlety

Servlety jsou základním technologickým prvkem Java webových aplikací, které poskytují jednoduché a konzistentní rozšíření webových serverů o aplikační logiku. Jedná se o prvky, které jsou volány pomocí HTTP požadavků a které následně provedou na serveru nějakou funkčnost nadefinovanou pomocí programovacího jazyku Java a vrátí standardní HTTP odpověď. [12]

Po technické stránce je Servletem jakákoliv třída, která dědí ze třídy `javax.servlet.http.HttpServlet`. Základními metodami jsou `protectedvoiddoGet(HttpServletRequestrequest, HttpServletResponse response)` a `protectedvoiddoPost(HttpServletRequestrequest, HttpServletResponse response)`, které se provedou po zavolání GET, resp. POST požadavku.

4.1.2 Filtry

Dalším prvkem webových aplikací v Javě jsou filtry. Jedná se o komponentu, která dynamicky zachycuje požadavky a odpovědi za účelem provedení definované funkčnosti nebo transformace. Jejich účelem není poskytovat odpověď, ale univerzální funkčnost. Konkrétním případem může být například doplnění nějakého parametru do požadavku. Příklady, kdy se filtry používají, můžou být:

- autentizace – přístup k obsahu pouze pro dané uživatele,
- logování – sledování přístupu k jednotlivým požadavkům/stránkám,
- konverze obrázků – úpravy měřítek a další,
- komprese dat – např. při stahování souborů,
- lokalizace – zobrazení odpovědi v předvoleném jazyce uživatele,
- a další. [13]

4.1.3 Java Server Pages (JSP)

Technologie JSP stránek poskytuje zjednodušený a rychlý způsob jak vytvářet dynamický webový obsah. Jedná se o soubory s příponou *.jsp*, které v sobě kombinují HTML kód a Java kód.

Samotnému uživateli se stránka ve webovém prohlížeči zobrazí standardním způsobem pouze za pomoci HTML kódu, protože Java kód je proveden na serveru a v odpovědi se již nevrací. [14]

Části, které jsou napsány pomocí Java kódu, jsou označovány jako skriptlety a jejich základní syntaxí je blok klasického Java kódu, který je ohraničen těmito znaky: `<% %>`. Takovýto kód je vykonán okamžitě, když se k němu při zpracování stránky dojde. Další možností je Java kód ohraničit blokem `<%! %>`. V tomto bloku jsou nadefinované třídy, metody a proměnné, které poté můžeme v rámci JSP stránky používat. Třetím nejrozšířenějším blokem je `<%= %>`. Uvnitř takto ohraničeného bloku se nachází nějaký výraz, který je potom vložen do HTML stránky. Konkrétně si můžeme představit například volání nějaké metody, která vrací hodnotu nebo samotnou proměnnou, jejíž hodnota je poté vložena do stránky.

Výhodou JSP stránek je i to, že je lze do sebe různě začleňovat pomocí notace `<%@include %>`. To nám umožňuje např. vytvořit základní stránku, která se bude chovat jako šablona a do které poté budeme začleňovat další JSP stránky jako kontejnery.

4.1.4 JSTL notace

JSTL notace je rozšířením JSP stránek o použití značkovacího jazyka poskytující základní funkčnost, kterou bychom mohli definovat i pomocí Java kódu ve skriptletech. Celým názvem zkratka JSTL znamená JavaServerPages Standard TagLibrary a jednotlivé značky lze rozdělit do těchto skupin:

- CoreTags,
- FormattingTags,
- SQL Tags,
- XML Tags,
- JSTL Functions. [15]

Například ve skupině CoreTags jsou značky, které nám umožní nadefinovat základní funkční bloky, jako jsou rozhodovací prvky if-else, cykly, odchytávání výjimek pomocí catch, výpis daného výrazu a další.

Abychom mohli tyto prvky používat, tak webová aplikace musí mít přístup k odpovídající knihovně, typicky pomocí jar souboru, který je buď přímo v aplikačním serveru, nebo u naší webové aplikace. Druhou podmínkou je definice knihovny v JSP stránce pomocí notace `<%@ taglib %>`.

4.1.5 Konfigurační soubor web.xml

Pro korektní běh webové aplikace v Javě je potřeba nakonfigurovat několik věci. Pro tento účel se používá hlavní konfigurační soubor *web.xml*, korektním pojmenováním tohoto souboru je DeploymentDescriptor. Jak již přípona naznačuje, jedná se o soubor, který je definovaný pomocí XML značkovacího jazyka.

Pomocí tohoto souboru lze nadefinovat spoustu věcí, ale ty hlavní vzhledem k výše uvedeným technologiím jsou:

- definice Servletu a namapování jeho třídy na konkrétní cestu v URL,
- definice Filtrů a jejich namapování na konkrétní požadavky,

- definice domovského souboru webu (tzv. welcomefile),
- definice chybových stránek, které se mají zobrazit po vrácení standardního HTTP chybového kódu,
- a mnoho dalších.

4.2 Aplikační server GlassFish

V rámci praktické částí této bakalářské práce jsem využíval aplikační server GlassFish, a proto jsem se rozhodl jej více přiblížit.

Jedná se o open source aplikační server sloužící pro platformu Java EE, který vyvíjela společnost Sun Microsystems. Po zániku společnosti Sun Microsystems přešel vývoj pod společnost Oracle a server má nyní celý název OracleGlassFish Server. [16]

Jedná se o referenční implementaci pro Java EE, a to znamená, že jako takový, server podporuje veškeré prvky platformy Java EE, mezi které patří EnterpriseJavaBeany, Java Persistence API, JavaServerFaces, JavaServerPages, Servlety, JMS, RMI a další.

Z pohledu operačního systému se jedná o multiplatformní aplikaci, existují tedy verze pro různé druhy operačních systémů. Poprvé byl GlassFish představen 6. června roku 2005 a současnou verzí je verze 4.1, která byla vydána 9. září 2014.

Výhodou a také hlavním důvodem, proč jsem se rozhodl použít tento aplikační server je jeho integrace ve vývojovém nástroji NetBeans, kde je server nainstalován již spolu s vývojovým prostředím a je možné prakticky okamžitě začít spouštět vyvíjené webové aplikace bez nutnosti dalšího složitého nastavení a konfigurace.

5 Databáze

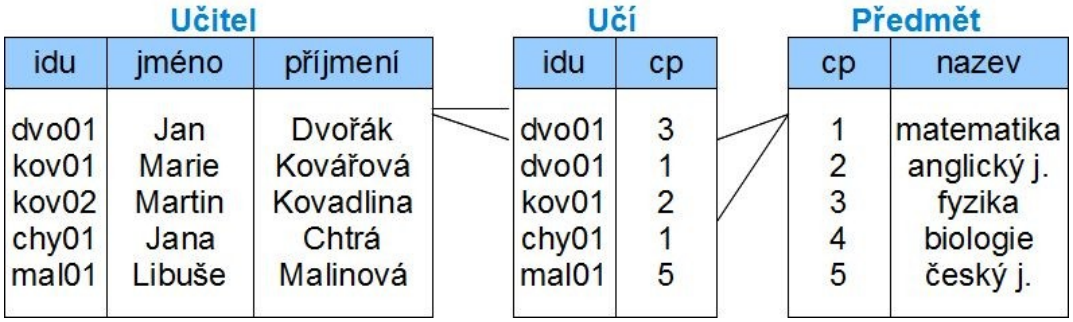
Databáze slouží k uchovávání dat a práci s nimi. Data jsou uložena v externím souboru, často i na externím disku a po restartu programu mohou být znovu načtena. Ke komunikaci s databází je využíván jazyk SQL. Základními dvěma typy databází jsou relační databáze a objektové databáze. Nejčastěji se používají relační databáze.

5.1 Relační databáze

Relační databáze byla poprvé definována v roce 1970 Edgarem F. Coddem. Již podle názvu jsou relační databáze založeny na relacích a jejich definice zní: „Relační databází v daném časovém okamžiku je konečná množina relací R1, R2, ..., Rm, tzv. aktuálních relací, kde Ri je typu Ri.“ [23] Definice relace poté zní takto: „Relace R s relačním schématem R je konečná podmnožina kartézského součinu domén Di, příslušejících jednotlivým atributům Ai“. [23] Zbývá ještě definice relačního schématu, která zní: „Relační schéma R je výraz tvaru R(A, f), kde R je jméno schématu, A = {A1, A2,..., An} je konečná množina jmen atributů, f je zobrazení přiřazující každému jménu atributu Ai neprázdnou množinu (obor hodnot atributu), kterou nazýváme doménou atributu Di, tedy f(Ai) = Di“. [23]

Zjednodušeně si můžeme relační databáze představit jako sadu tabulek, které jsou propojeny předem nastavenými vztahy. Tabulka se skládá ze sloupců (atributů) a z řádků (záznamů). Atributy tabulek určují vlastností jednotlivých objektů, které se budou v tabulce ukládat. Vlastností tabulek je, že na pořadí řádků a sloupců nezáleží, ale pro to, aby bylo možno jednoznačně identifikovat řádek tabulky, se používají primární klíče. Primární klíč může být složen buď z jednoho, nebo více atributů a musí tedy být jednoznačný – tzn., že pokud mají dva záznamy stejný primární klíč, jedná se o stejný záznam. Dalším speciálním atributem, který umožňuje vytvořit vazby mezi tabulkami je cizí klíč. Do takového atributu se poté ukládá primární klíč jiné tabulky.[25]

U vazeb mezi tabulkami se definují dva pojmy. Prvním je povinnost vazby. Tato povinnost určuje, zda musí daný objekt mít vazbu na jiný objekt. Např. u vazby zaměstnanec – telefon povinná vazba není, protože ne každý zaměstnanec musí mít telefon. Druhým pojmem je kardinalita. Ta určuje mohutnost vazby, neboli kolik objektů bude ve vazbě vystupovat. Existují tři základní kardinality: 1:1, 1:N a M:N. Vazba 1:1 znamená, že jeden objekt má přidružen maximálně jeden druhý objekt. V našem příkladu u zaměstnanců a telefonů by to znamenalo, že jeden zaměstnanec může mít pouze jeden telefon. Vazba 1:N říká, že jeden objekt může mít přidružených více druhých objektů. Na příkladu by to tentokrát bylo, že jeden zaměstnanec může mít více telefonů. Nejobecnější vazbou je vazba M:N, která říká, že více objektů může mít ve vazbě více druhých objektů – více zaměstnanců sdílí více telefonů. Vazbu M:N jako jedinou nemůžeme přímo v databázi ukládat a je potřeba ji rozložit pomocí vazební tabulky na dvě vazby 1:N. Ukázka rozkladu ve vazbě Učitel UČÍ Předmět je na obrázku 5.1.



Obrázek 5.1: Ukázka rozkladu vazby M:N [25]

5.2 Java DB

Java DB je relační databáze, která je založena na programovacím jazyce Java a na SQL. Jedná se o open source databázi Apache Derby, kterou pod svou hlavičkou vydala společnost Oracle. Derby je zde začleněno bez změn ve zdrojovém kódu. Tato databáze je součástí JDK od verze 6. [17]

První verze databáze Apache Derby byla vydána v roce 1997 pod záštitou společnosti Cloudscape Inc. Původním názvem bylo JBMS. Následně byla databáze přejmenována na Cloudscape. V roce 2001 vývoj databáze převzala společnost IBM, která jej poté v srpnu roku 2004 předala pod uskupení Apache Software Foundation. Zde byla databáze poprvé pojmenována Derby. K vývoji se poté přidala společnost Sun a následně, již pod značkou Oracle, byla databáze začleněna do JDK pod názvem Java DB.[18]

Databáze podporuje dva módy práce. Tím prvním je mód server, kdy databáze běží samostatně a obsluhuje veškeré požadavky, které jsou na ni poslány. Druhým módem je mód embedded, kdy databáze běží v JVM pouze v rámci dané aplikace, tzn. že ostatní aplikaci k této databázi nemohou přistupovat.

Výhodou této databáze je její nízká velikost a přenositelnost. Pro přístup k databázi jsou podporovány všechny standardní způsoby jako je JDBC, JPA, Hibernate nebo JDO. Další výhodou je její integrace ve vývojovém nástroji NetBeans, což je opět také hlavní důvod, proč jsem se rozhodl tuto databázi použít.

5.3 Přístup k databázi v Javě

Jak už bylo zmíněno výše, pro práci s databází se standardně využívá jazyka SQL. Příkazy v tomto jazyce potom zadáváme pomocí nějakého klientského programu pro přístup ke konkrétní databázi. Takovým klientským programem může být konzolová aplikace běžící pod Linuxem nebo i plné grafické prostředí pro administraci databáze.

Pokud chceme k databázi přistupovat z naší aplikace pomocí Java kódu, potřebujeme k tomu jistou knihovnu nebo funkce, které zajistí připojení na databázi a komunikaci s touto databází. V Javě máme čtyři základní možnosti, jak k databázi přistupovat:

- pomocí JSP – využívá se JSTL značek ze skupiny SQL Tags,
- pomocí JDBC ovladače,

- pomocí hotového cizího frameworku,
- pomocí objektově-relačního mapování (ORM). [19]

První případ pomocí JSTL značek je sice asi nejjednodušší, ale neumožňuje nám vložit žádnou aplikační logiku – výsledek volaných SQL dotazů je přímo zobrazen do prezentační vrstvy. Druhý případ pomocí JDBC ovladače nám umožňuje vytvořit si vlastní vrstvu pro přístup k databázi (tzv. DAO vrstvu). Tuto vrstvu si můžeme představit jako sadu tříd, ve kterých budeme s databází pracovat. U třetí možnosti (hotového frameworku) většinou již existuje nějaká šablona nebo hotové funkce pro práci s databází. Na pozadí je poté opět většinou JDBC. Příkladem takového hotového frameworku může být třeba Spring. Poslední možnost, přístup pomocí ORM, je asi nejškálovatelnější a nejabstraktnější řešení. Můžeme si napsat vlastní ORM, což nám umožní mít plnou kontrolu nad prací s databází, ale na úkor velké práce a rozsáhlého kódu. Další možností je pak využít nějaké hotové řešení, které v Javě implementuje Java Persistence API. Takovým hotovým ORM řešením může být např. Hibernate, EclipseLink a další. Při využití hotového ORM se již většinou nepoužívá čistý SQL jazyk, ale většinou jeho upravená verze. Pro Hibernate je to např. jazyk HQL.

5.3.1 JDBC

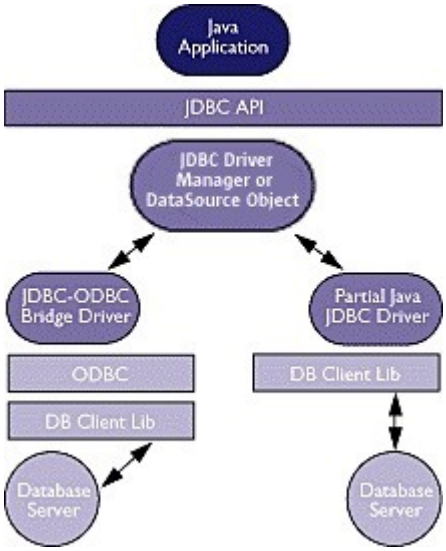
V praktické části této bakalářské práce využívám přístup k databázi pomocí JDBC ovladače. Toto řešení jsem vyhodnotil pro mé potřeby jako nejefektivnější, a to z důvodu, že mnou využívaná databáze není příliš rozsáhlá a vytvářet vlastní ORM by bylo zbytečně náročné.

Celým názvem zkratka JDBC znamená Java Database Connectivity. Jedná se o API pro programovací jazyk Java, které definuje přístup klienta k databázi. JDBC poskytuje podporu pro velikou škálu databází s jednotným rozhraním pro programátora. To znamená, že námi naprogramovaný kód v Javě bude stejný bez ohledu na to, jestli budeme pracovat s databází MySQL, Oracle nebo nějakou úplně jinou. Jediné, co se bude měnit, bude použitý JDBC ovladač. [20]

Základními funkcemi, které JDBC poskytuje, jsou:

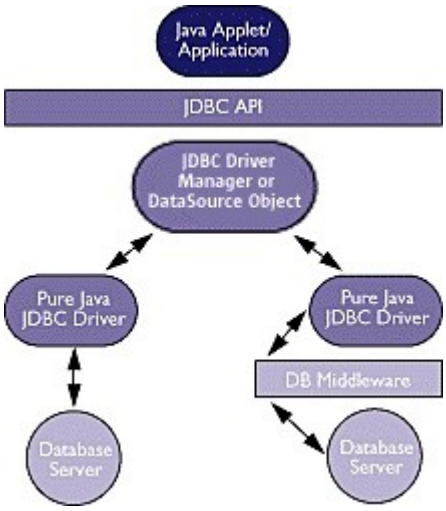
- vytvoření spojení na databázi,
- posílání SQL příkazů,
- zpracování výsledků.

Architektura JDBC lze rozdělit do 4 skupin podle typu. První typem je most JDBC-ODBC. Tato kombinace poskytuje JDBC přístup pomocí ODBC ovladačů. Binární kód ODBC musí být nahrán na každém klientském stroji, který chce spojení využívat. Další nevýhodou je také nižší výkon, protože se využívá dvou ovladačů. Princip tohoto typu lze vidět na Obrázku 5.2 v levé části. Druhým typem je nativní API. Tento typ ovladače využívá z části Java jazyk a z části nativní jazyk. Jeho principem je překládání volání JDBC na volání klientského API pro jednotlivé databáze. Nevýhodou je, že podobně jako v předchozím případě, i zde je potřeba, aby na každém klientském stroji byl nainstalován příslušný software. Princip lze opět vidět na Obrázku 5.2, tentokrát v pravé části. [20]



Obrázek 5.2: První dva typy architektury JDBC [20]

Třetím typem je ovladač využívající aplikační server. JDBC ovladač je napsán kompletně v Javě a dotazy posílá na aplikační server, kde běží serverová část tohoto ovladače, která požadavky přeloží pro konkrétní databázi. Výhodou je, že při změně databáze není nutný žádný zásah do klientské části, pouze se na serveru změní odpovídajícím způsobem serverová komponenta. Princip je vidět na Obrázku 5.3, v pravé části. Posledním, čtvrtým, typem ovladače je přímý přístup ovladače k databázi. Už podle názvu je zjevný jeho princip. Jedná se o ovladač kompletně napsaný v Javě, který sám převádí volání požadavků na přímé volání databáze přes síťovou vrstvu. Jedná se asi o nejjednodušší typ ovladače, který se používá nejčastěji. Takovýto ovladač bývá součástí vyvíjené aplikace a nepotřebuje již žádný další software ani na klientském, ani na serverovém stroji. Jeho princip je také znázorněn na Obrázku 5.3 vlevo. [20]



Obrázek 5.3: Další dva typy architektury JDBC [20]

Co se týče historie, tak JDBC bylo poprvé vydáno společnosti Sun Microsystems 19. února 1997 jako součást JDK 1.1. Nejnovější verzí je verze 4.2, která vyšla 5. prosince 2013. Při výběru JDBC ovladače je důležité zjistit, kterou verzi podporuje námi používaná databáze. Co se týče implementace, tak veškeré třídy, které JDBC využívá, jsou obsaženy v balíčcích java.sql a javax.sql. [21]

Ukázka vytvoření připojení k databázi:

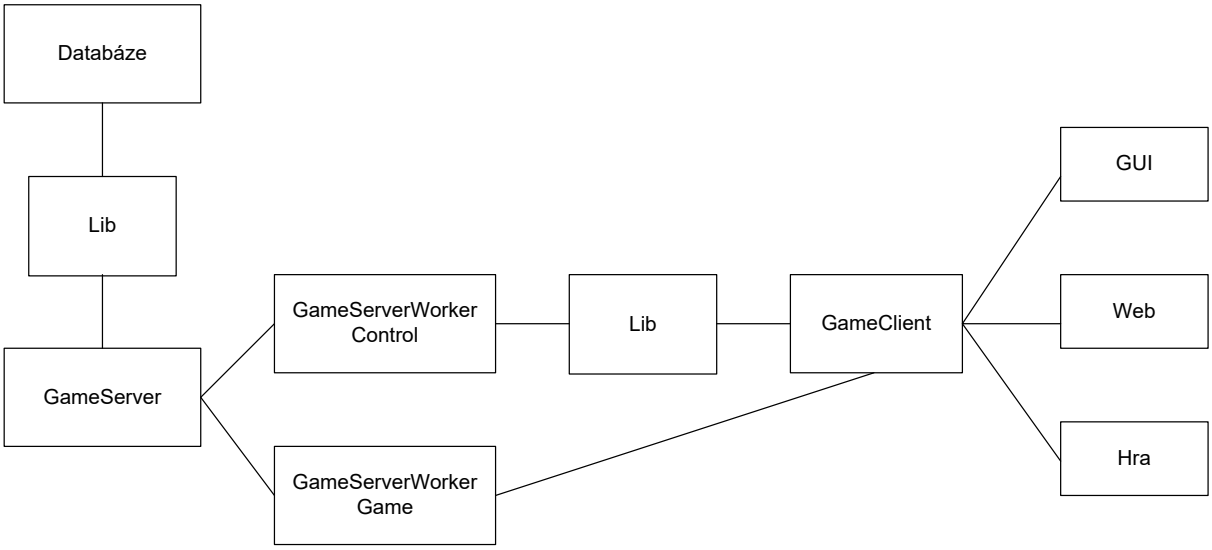
```
String host = "jdbc:derby://localhost:1527/GameServerDatabase";
String username = "a";
String password = "a";
Connection con = DriverManager.getConnection(host, username,
password);
```

Ukázka dotazu provedeného nad databází, konkrétně získání hráče podle loginu:

```
Statement stmt = con.createStatement();
String sql = "SELECT * FROM Player WHERE Login =
'".concat(login).concat("'");
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()){
      player.setId(rs.getInt("ID"));
      player.setLogin(rs.getString("Login"));
      player.setPassword(rs.getString("Password"));
      player.setJmeno(rs.getString("Jmeno"));
      player.setPrijmeni(rs.getString("Prijmeni"));
      player.setEmail(rs.getString("Email"));
}
```

6 Popis implementace

Celý herní portál je rozdělen do několika částí, jeho zjednodušenou strukturu si ukážeme na obrázku 6.1. Všechny tyto části popíšeme v následující kapitole.



Obrázek 6.1: Struktura herního portálu

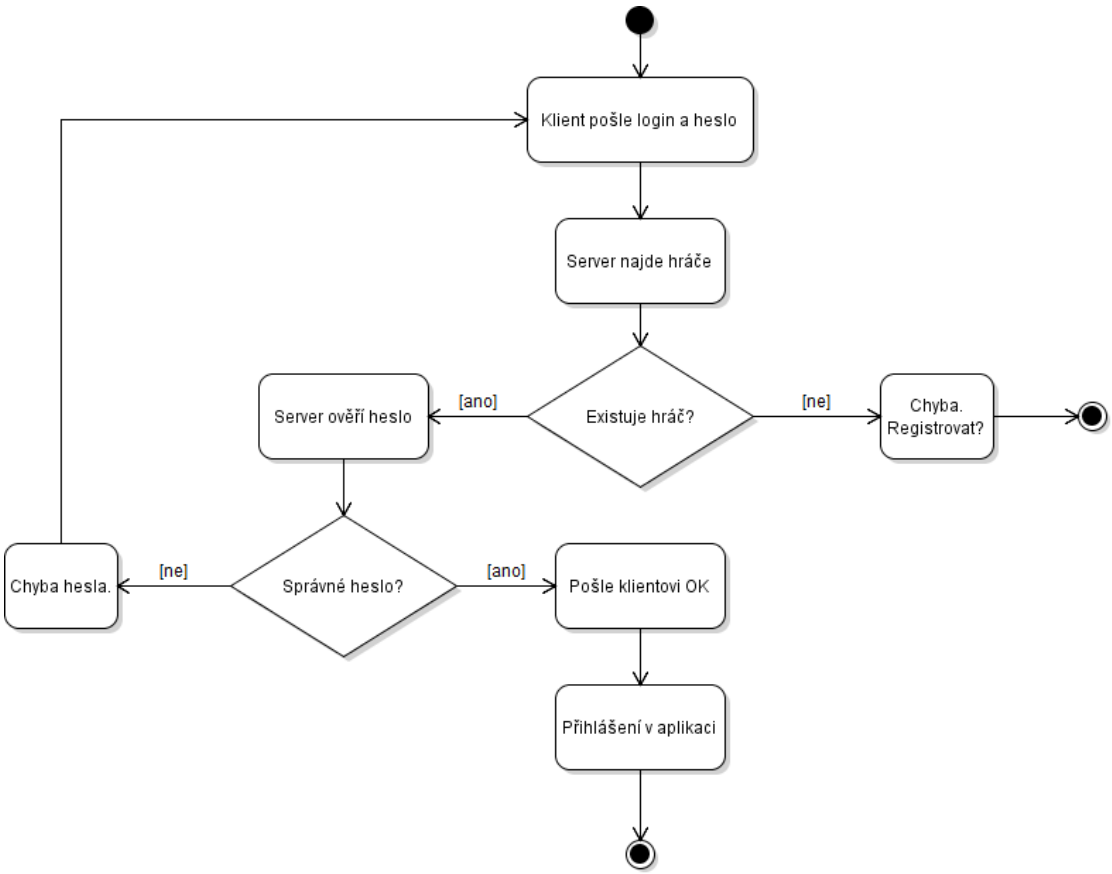
6.1 Server

Základem herního portálu je server. Ten by měl být pořád zapnutý. Pro vytvoření herního serveru byl použit návrhový vzor singleton. To nám zajišťuje, že server bude vždy jenom jeden a nemělo by docházet ke ztrátám. Server má dvě základní třídy: `GameServer`, `GameServerWorker`.

`GameServer` je hlavní třída, ve které jsou uloženy všechny údaje, a která celý portál obsluhuje. Ve třídě `GameServer` jsou v kolekcích uloženy údaje o hráčích, hrách, turnajích a přihlášených hráčích. Všechny údaje jsou uloženy v kolekci typu `List`. Veškeré údaje se ukládají také do databáze, protože když je nutno server z nějakého důvodu vypnout, o všechny data bychom přišli. Při spuštění serveru se zase všechny data načtou z databáze a uloží do listů. Server také obsahuje metody pro zisk konkrétního údaje z listu nebo ukládání nových údajů do listu, případně odstranění určitého údaje z kolekce. Součástí serveru je i plánování časových her za pomoci `TimerTasku`. Po spuštění se kromě načtení dat z databáze také vytvoří `ServerSocket` a obslužná třída.

Obslužná třída`GameServerWorker` slouží k navázání spojení a komunikaci s klientem přes socket. Hlavní funkcí je zpracování příkazů od klienta za pomoci serveru. Třída `GameServerWorker` může být dvojího typu: `Control` nebo `Game`. Vytváří se při spuštění serveru a vyčkává na socket od klienta. Když tento socket obdrží, obdrží zároveň i informaci o tom jakého má být typu, při tomto postupu je to vždy typ `Control`.

`GameServerWorker` typu `Control` se stará o všechny funkce serveru a komunikaci s uživatelem. Když získá typ `Control` je bez třídního atributu `PlayerLog`. Jediná funkce, která je umožněna v tomto okamžiku klientovi, je login a register. Funkci login si nyní ukážeme podrobněji na aktivitním diagramu na obrázku 6.2.



Obrázek 6.2: Aktivitní diagram funkce login

Klient se chce přihlásit, vyplní tedy údaje a klikne na tlačítko login. Tyto údaje se uloží do socketu a pošlou na server, kde je obslužná třída získá ze socketu. V této třídě je metoda login, která zjistí, zdali je uživatel pod tímto jménem uložen.Pokud ano, tak zkontroluje, jestli zadal správné heslo. Kontrola se děje tak, že se zavolá metoda třídy `GameServer`, která projde kolekci registrovaných hráčů a podá obslužné třídě informace o daném hráči, případně informaci, že daný hráč neexistuje. Pokud vše proběhne v pořádku, tak obslužná třída získá třídní atribut `loggedPlayer` a pošle serveru informaci, aby si tohoto hráče přidal do kolekce přihlášených hráčů. Klientovi se přes socket pošle zpráva, že daný uživatel byl přihlášen.

Funkce register je obdobná, s tím rozdílem, že uživatel není v databázi a musí vyplnit i jméno, příjmení a e-mail. Pokud vše proběhne v pořádku, tak server uživatele uloží do databáze, vlastní kolekce a rovnou jej přihlásí.

Jakmile se uživatel přihlásil, získal tím `GameServerWorker` třídní atribut `PlayerLog` a odemknul uživateli všechny funkce. Mezi tyto funkce patří podívat se na hráčův profil. Když přijde

tento požadavek `GameServerWorker` pošle požadavek serveru a ten mu z databáze získá informace o hráči a jím odehraných hrách. Obslužná třída tyto údaje potom pošle přes socket klientovi k dalšímu zpracování.

Pokud přijde přes socket požadavek na vytvoření nového turnaje,obslužná třída řekne serveru, aby si tyto údaje uložil do své kolekce i databáze a vytvořil novou instanci třídy `Tournament`. Pokud vše proběhne v pořádku tak se o tomto turnaji pošlou informace klientovi. Turnaje jsou dvojího typu. Buď tabulka, tzn. každý s každým a vítězové dostávají body, nebo pavouk, čili klasický vyřazovací turnaj. U turnajů se musí nastavit počet hráčů a datum kdy začne první kolo. Turnaj typu tabulka může být spuštěn pro 2 až 200 hráčů. Vyřazovací turnaj typu pavouk lze nastavit pro 4, 8, 16 nebo 32 hráčů.

Nová hra se vytváří obdobně jako turnaj, akorát místo třídy `Tournament` se vytváří třída `GameInstance`. Nová hra se dá vytvořit, kromě okamžitého spuštění, i na spuštění v určitou hodinu a den.

Při přidávání hráčů do turnaje nebo hry získá obslužná třída serveru informaci o hráči a id dané hry nebo turnaje od socketu. Server zjistí, jestli je ještě v turnaji či hře místo a pokud ano hráče do ní přidá. Do databáze je uložen záznam, že daný hráč hrál danou hru a informace o úspěšném přidání hráče jsou přes socket odeslány klientovi. Když je hráč přidán do hry, která je naplánovaná na určitý čas, může se v mezičase odhlásit a pokud se stihne znovu přihlásit, před začátkem naplánované hry, ta se mu automaticky v daný čas zapne.

Dále server nabízí možnost chatu. Pokud přijde požadavek se zprávou, obslužná třída se podle parametrů rozhodne, kterou funkci serveru zavolá. Na výběr jsou tři: poslat zprávu všem uživatelům, poslat zprávu hráčům, kteří hrají stejnou hru anebo jen jednomu konkrétnímu hráči.

Funkce logout je přesný opak funkce login. Ze serveru je daný hráč odstraněn z kolekce přihlášených hráčů. Ve třídě `GameServerWorker` je třídní atribut `PlayerLog` nastaven na null a klientovi je poslána zpráva, že uživatel byl odhlášen. K odhlášení může dojít ve třech případech:

- hráč stiskne tlačítko logout,
- hráč se přihlašuje, ale jiný hráč je již v tomto klientovi přihlášen a proto bude odhlášen,
- uživatel zavře klienta nebo jiným způsobem dojde ke ztrátě spojení.

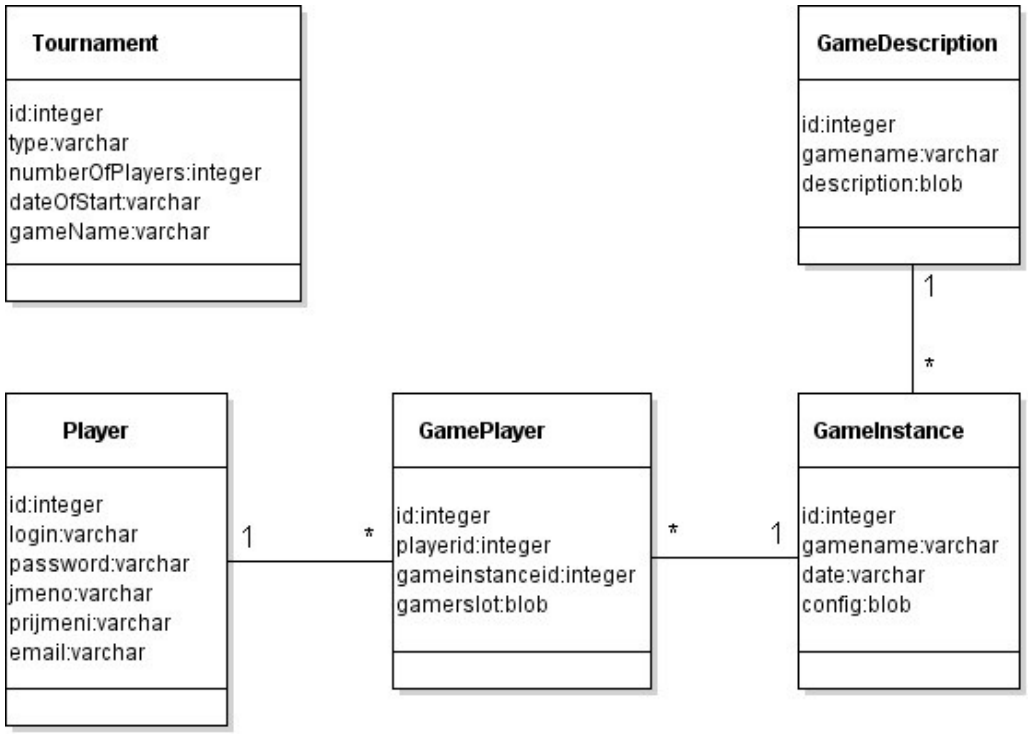
Druhým typem třídy `GameServerWorker` je typ `Game`. Vytváří se při spuštění hry a stará se pouze o běh hry. Všechny ostatní příkazy, např. novou zprávu do chatu řeší typ `Control`, klient má tím pádem v daný moment dvě instance třídy `GameServerWorker`. Jedna se stará o běh hry a druhá o funkcionalitu programu.

6.2 Databáze

Databáze v tomto programu slouží jako záloha dat. S databází pracuje pouze třída `GameServer` za asistence pomocných tříd uložených v knihovně. Při spuštění serveru se načítají všechny údaje o hráčích, hrách a turnajích právě z databáze a server si je ukládá do vlastní kolekce, aby s nimi mohl dále pracovat. Do databáze se ukládá vždy, když server vytvoří něco nového. Mezi hlavní tabulky patří:

- Player - ukládá údaje o hráči jako je jeho login, jméno, email atd.,
- Tournament - zde se nachází údaje o turnajích např.:typ turnaje, počet hráčů atd.,
- GameDescription - tady jsou uloženy údaje o hrách např.: jméno, minimální věk atd.,
- GameInstance - v této tabulce jsou údaje o jednotlivých instancích hry.

Součástí databáze je i vazební tabulka, do které se vkládají záznamy, když se hráč přidá do nějaké hry. Tato tabulka je také základem pro zobrazení hráčského profilu a informací o odehraných hrách. Strukturu této databáze si ukážeme na obrázku 6.3.



Obrázek 6.3: Struktura databáze

6.3 Klient

Druhou nejvýznamnější částí herního portálu je klient. Klient se skládá ze dvou hlavních částí a to třídy `GameClient`, která komunikuje se serverem pomocí socketů a uživatelského rozhraní.

Třída `GameClient` při startu naváže přes socket spojení se serverem. Plnění jednotlivých funkcí funguje obdobně. Třída získá data od uživatele pomocí uživatelského rozhraní a pošle je ke zpracování na server. Jakmile obdrží data od serveru, tak si je buď přidá do třídních atributů, anebo je pošle uživatelskému rozhraní k zobrazení.

Funkce login a register jsou obdobné. Klient vyplní login a heslo, v případě registrace i další nezbytné údaje a ty jsou pak socketem poslány na server. Pokud klient dostane odpověď, že vše proběhlo v pořádku, tak se nastaví třídní atribut `PlayerLog` na daného uživatele a odemknou se ostatní funkce. Pokud vše neproběhne v pořádku, objeví se chybová hláška o tom, co se stalo, např. že zadaný login neexistuje apod.

Vytváření nových her a turnajů funguje opět podobně. Uživatel musí vyplnit všechny údaje, jako např. počet hráčů, jestli má hra začít hned nebo v určitý čas a v jaký čas, atd. Všechny údaje jsou poté poslány na server, který hru či turnaj vytvoří a klientovi přijde zpráva, že byla vytvořena nová hra nebo turnaj.

Funkce chat má tři parametry, které se nastavují podle toho, kterou možnost označí hráč v uživatelském rozhraní. Tyto parametry jsou:

- message - je vyplněn vždy a kromě vlastní zprávy obsahuje i login hráče, který zprávu posílá,
- gameName - je vyplněn pokud má být zpráva odeslána pouze uživatelům, kteří hrají stejnou hru(jinak je null),
- login - je vyplněn pokud má být zpráva poslána pouze tomuto hráči(jinak je null).

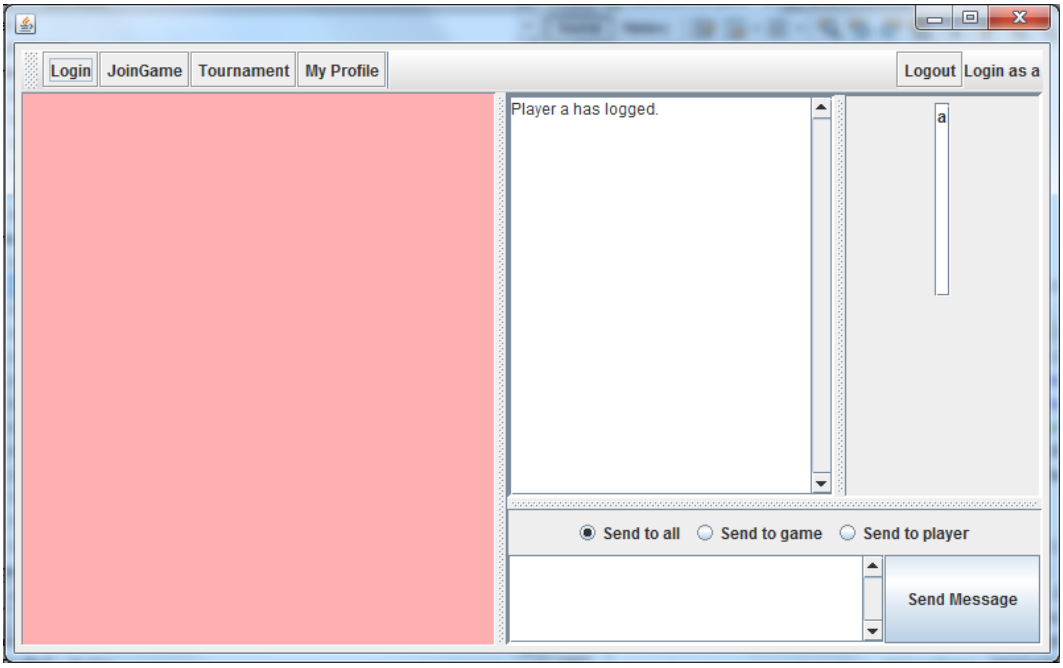
Podle těchto parametrů server pozná, komu má zprávu poslat. Pokud je gameName i loginnull, tak pošle zprávu všem uživatelům. Klient obdrženou zprávu pouze zobrazí.

Mezi další funkce klienta patří získávání různých údajů od serveru, které jsou poté zobrazeny uživateli.

Funkce logout slouží k odhlášení uživatele. Serveru je poslána zpráva, že se daný uživatel chce odhlásit a ten jej odhlásí. Klientovi poté přijde zpráva, že byl uživatel odhlášen.

6.3.1 Uživatelské rozhraní

Uživatelské rozhraní slouží k celkové obsluze programu. Pomocí uživatelského rozhraní se načítají vstupy od uživatele a zobrazují zpracované výsledky, které nám poslal server. Celá obrazovka je po zapnutí rozdělena do tří hlavních částí. V horní části je tlačítkové menu, vlevo dole se nachází část pro hry a vpravo dole je chat. Uživatelské rozhraní si ukážeme na obrázku 6.4.



Obrázek 6.4: Uživatelské rozhraní

Část věnovaná hrám je po spuštění prázdná. V této části se nic kromě běhu hry neděje. Tato část se skládá pouze z prvku `JPanel`, ve kterém běží hra.

Část, ve které se nachází chat, je rozdělena do dvou dalších částí. V horní polovině se nachází seznam přihlášených uživatelů. Z tohoto seznamu lze vybrat vždy jen jednoho uživatele, kterému může být poslána zpráva. Vedle tohoto seznamu se nachází prvek `JScrollPane`, ve kterém se zobrazují příchozí zprávy. Tento prvek je nastaven tak aby se při velkém počtu zpráv posouval na aktuální zprávu. `JScrollPane` byl zvolen proto, aby se v něm dalo posouvat i na starší zprávy.

V dolní části se nacházejí tři prvky `JRadioButton`, které slouží k výběru, komu bude zpráva poslána. Dále je zde prvek `JTextArea`, do kterého se píše nová zpráva, kterou chceme poslat. Napravo od tohoto prvku je tlačítko `JButton`, kterým se zpráva posílá.

Poslední částí je menu, složené z tlačítek. Každé z tlačítek otevírá nové okno, kromě posledního tlačítka, které plní funkci logout. Vedle tohoto tlačítka se nachází prvek `JLabel`, ve kterém se zobrazuje, jaký hráč je zrovna přihlášen, případně že není přihlášen nikdo.

První tlačítko je tlačítko Login, které otevře nové okno. V tomto novém okně má uživatel dvě možnosti, buď vyplnit login a heslo a přihlásit se, nebo kliknout na tlačítko Register. V tomto případě se uživateli otevře další okno, ve kterém vyplní všechny údaje a registruje se.

Dalším tlačítkem je JoinGame. Po kliknutí se otevře nové okno, které umožňuje přihlašování do hry nebo tvorbu nové hry. V levé části se nachází seznam již vytvořených her, do kterých je možno se přihlásit. Pod tímto seznamem je tlačítko Refresh, které tento seznam aktualizuje. Napravo od tohoto tlačítka je další tlačítko, které slouží k přidání hráče do vybrané hry. Posledním tlačítkem ve spodní části je tlačítko Cancel, které nás vrátí na hlavní obrazovku. Vpravo se nachází tlačítka pro tvorbu nové hry a nové časované hry. Také je zde tlačítko na zobrazení detailů o vybrané hře.

Napravo od tlačítka Join Game se nachází tlačítko Tournament. Toto tlačítko otevírá velice podobné okno tomu předchozímu. Toto okno se však netýká her, ale turnajů. To znamená, že zde nalezneme seznam turnajů, tvorbu nového turnaje atd.

Posledním tlačítkem je My Profile. Toto tlačítko zobrazuje profil hráče, což jsou jeho údaje, které zadal při registraci. Dále se na této stránce zobrazují údaje o hráčem odehraných hrách. Tyto údaje jsou zobrazeny v tabulce.

6.4 Webové rozhraní

Celý web běží na GlassFish serveru. Web se skládá z JSP stránek a servletů. JSP stránky se starají o správné zobrazení dat a servlety tyto data získávají a starají se o správnou funkčnost. Web komunikuje se serverem přes sockety pomocí třídy `GameClient`. Při spuštění webu jsou povoleny jediné 2 možnosti a to login nebo register. Když se uživatel přihlásí případně zaregistruje tak se vytvoří instance třídy `GameClient`, která komunikuje se serverem a servlety přes tuto třídu posílají

své požadavky a získávají data. Data poslaná serverem jsou uložena ve třídě `GameClient` a servlety pouze tuto třídu požádají o zrovna potřebné údaje k zobrazení.

Při spuštění webu se zobrazí úvodní stránka a v hlavičce odkaz na stránku s loginem. Na této stránce je také možnost registrace. Pokud se uživatel přihlásí tak se hlavička změní a nabízí možnosti prohlídky hráčského profilu a logout. Na stránce profilu jsou zobrazeny údaje o hráči a jeho odehrané hry. Dále jsou zde možnosti pro úpravu libovolného údaje z hráčova profilu, případně smazání celého hráče. Po logaci se kromě hlavičky zobrazí stránka s výběrem hry. Když si hráč vybere hru zobrazí se mu nabídka, zda se chce dozvědět nějaké informace o hře, vytvořit novou standartní hru pro dva hráče nebo jestli chce založit nový turnaj. Turnaje lze vytvořit stejně jako v desktopové aplikaci. Uživatel si zvolí jestli chce typ turnaje každý s každým nebo vyřazovací turnaj. Nastaví počet hráčů a čas kdy má turnaj začít. Stránka s informacemi o hře obsahuje název hry, popis hry, pravidla hry, počet bodů a hráčů a obrázek ze hry.

6.5 Knihovna

V knihovně jsou uloženy pomocné třídy, které přímo nesouvisí ani s jednou dříve uvedenou kategorií. Třídy v knihovně můžeme rozdělit do několika skupin.

Nachází se zde 2 skupiny tříd pro komunikaci přes sockety. Každý požadavek ze strany klienta má vlastní třídu a najdeme je v balíčku `lib.server`, protože tyto požadavky zpracovává server. Každá tato třída implementuje rozhraní `ServerCommand`. Každá odpověď, kterou server posílá klientovi nebo dotaz na uživatele od serveru má také svou třídu. Tyto třídy jsou uloženy v balíčku `lib.client` a zpracovává je klientská část, zejména pak třída `gameClient`. Všechny tyto třídy rovněž implementují rozhraní `ClientCommand`.

Další skupinou jsou třídy, které nahrazují datové typy. Jsou využívány jak serverem, tak klientem. Mezi tyto třídy patří například `Player`, `PlayerLog`, `Tournament`, atd. Najdeme je v balíčku `lib.data`.

V balíčku `lib.database` je další skupina tříd, která se stará o práci s databází. Mezi tyto třídy patří: `RegisterPlayer`, `SaveGame`, `SaveGameDescription`, `SaveGameInstance` a `SaveGamePlayer`. Tyto třídy umí ukládat data do databáze a případně tyto data zobrazit nebo předat serveru. Třída `RegisterPlayer` navíc umožňuje vyhledat hráče, upravit profil hráče, vymazat hráče z databáze a nastavit hráči id.

Dalším balíčkem je `lib.game`, kde se nacházejí rozhraní, které musí daná hra implementovat, aby správně fungovala. Jsou zde také pomocné třídy a to `GameConnectionService` a `GamerSlot`, které se starají o obsluhu a chod hry.

Poslední částí knihovny je balíček nazvaný pouze `lib` a jsou v něm třídy, které se nehodí do žádné z předchozích kategorii. Součástí tohoto balíčku jsou třídy pro obsluhu turnajů, id generátor nebo třeba výčtový typ pro připojení.

6.6 Instalace nové hry

Pokud chce programátor spustit novou hru, musí tato hra rozšiřovat všechna rozhraní, která se nachází v knihovně lib.game. Mezi tyto rozhraní patří:

- GameConfig - údaje o nastavení konkrétní hry,
- GameDescription - popis hry,
- GameInstance - běh hry, přiřazování hráčů,
- GameInstanceDescription - získávání informací o dané instanci hry.

Dále musí být `GameDescription` této hry ručně přidán do kolekce ve třídě `GameServer`.

Rozhraní `GameConfig` je v podstatě prázdné rozhraní tak aby vyhovovalo všem druhům her. Jediný parametr, který je povinný pro všechny hry je počet hráčů. Toto rozhraní se používá jako nastavení jednotlivých her daného typu.

`GameDescription` je v podstatě popis hry, který musí být ručně přidán do kolekce ve třídě `GameServer`. Součástí tohoto rozhraní je id, název hry, popis hry, pravidla hry, hrací doba, doporučená minimální a maximální věková hranice a také minimální a maximální počet hráčů, kteří si můžou najednou zahrát.

`GameInstance` slouží k běhu hry. Aby hra správně fungovala musí rozšiřovat toto rozhraní a všechny jeho metody. Mezi ty hlavní patří připojení hráče do hry, zjištění zda-li je již hra dohraná, případně kdo je vítěz. Dále také získávání údajů o hře jako je bodový stav nebo screenshot. Toto rozhraní vytváří pomocnou třídu `GameConnectionService`, která řeší nahrazení hráče při ztrátě spojení, případně jeho znovu napojení do hry. K tomu využívá opět rozhraní `GameInstance` a jeho metody pošli bota, zabij bota a znovu připoj hráče.

`GameInstanceDescription` sdružuje všechny předchozí rozhraní. Slouží k získávání informací o dané instanci hry a k tomu využívá metod získej herní instanci, získej název hry, získej nastavení hry a získej údaje o hráčském slotu. Což je třída, která spojuje hráče s danou hrou.

6.7 Umělá inteligence

Když hra zaznamená ztrátu spojení s uživatelem, vyzve server k řešení situace. Ten požádá hru o umělou inteligenci (bota) a uloží si údaje o hráči, kterému se ztratilo spojení. Daného bota pošle hře k znovu připojení. V případě, že se hráč, který přišel o spojení znovu přihlásí v době, kdy hra ještě probíhá, bude dotázán zda-li se chce znovu připojit. Pokud ano tak server pošle hře informaci, že má zabít bota a znovu přihlásit daného hráče. Server si údaje o hráči smaže pokud se hra dohraje nebo pokud se uživatel přihlásí ještě v průběhu hry a to i v případě, že se znovu nezapojí do hry. O tuto funkčnost se stará třída `GameConnectionService`, kterou lze nalézt v balíčku `lib. game`.

Na serveru funguje administrátor, který může dávat bany hráčům. Každému kdo dostane ban je znemožněno se přihlásit jak na web tak do desktopové aplikace, a to do té doby než jim administrátor znovu povolí přístup. Tento administrátor má na webu speciální stránku, na kterou má přístup pouze on. Na této stránce má zobrazen list všech registrovaných hráčů a checkboxy, které mu umožní snadno udělit ban, případně ho zrušit.

7 Závěr

Cílem této bakalářské práce bylo vytvořit internetový herní portál v programovacím jazyce Java, který by umožňoval hraní her, které byly vytvořeny v rámci jiných bakalářských prací. Tento herní portál měl také podporovat správu uživatelů, jednotlivých her, vytváření turnajů, textovou konverzaci hráčů a časové plánování her. Herní portál měl být dostupný jak přes desktopovou verzi klienta, tak přes webové rozhraní.

Na začátku své práce jsem uvedl teoretické znalosti, které se týkaly zpracovaného programátorského řešení. Byly také popsány jednotlivé technologie jazyka Java, které jsou v této práci využity, princip tvorby webových aplikací a práce s databázemi v jazyce Java. Následně jsem popsal praktickou implementaci portálu, samotnou strukturu portálu a strukturu používané databáze. Detailně jsem také popsal funkčnost jednotlivých částí herního portálu. Popis praktické implementace jsem také doplnil o obrázky a diagramy jazyka UML.

Samotné programové řešení bylo vyvinuto ve vývojovém prostředí NetBeans v jazyce Java. Komunikace mezi klientem a serverem probíhá pomocí socketu. Webová část herního portálu běží na aplikačním serveru GlassFish a je vytvořena pomocí JSP stránek a servletů. Webová část aplikace se serverem také komunikuje pomocí socketů.

Vytvořený herní portál je funkční a zadání této bakalářské práce bylo dle mě splněno, s výjimkou bodů uvedených v úvodu této práce, kde je také zdůvodnění, proč byly tyto body vynechány. Osobním přínosem této práce pro mě bylo rozšíření znalosti programovacího jazyka Java a jeho technologií, vyzkoušení si vývoje většího projektu a také poznání vývoje webových aplikací.

Použitá literatura

[1] About the Java Technology. In: The Java™ Tutorials [online]. [cit. 2015-07-24]. Dostupné z: http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

[2] PECINOVSKÝ, Rudolf. Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky. 2., aktualiz. a rozš. vyd. Praha: GradaPublishing, 2009, 570 s. ISBN 978-80-247-2653-3.

[3] Java - rozhraní, polymorfismus, vyjímky [online]. [cit. 2015-07-24]. Dostupné z: http://portal.haka-software.cz/index.php?option=com_content&view=article&id=9:java-rozhrani-polymorfismus-vyjimky&catid=3:programovani-java&Itemid=13

[4] Java VirualMachine JVM tutorial [online]. [cit. 2015-07-24]. Dostupné z: http://viralpatel.net/blogs/java-virtual-machine-an-inside-story/

[5] Collections in Java [online]. [cit. 2015-07-25]. Dostupné z: http://learnwithharsha.com/day-5-collections-framework/

[6] 12fig01.gif [online]. [cit. 2015-07-25]. Dostupné z: http://flylib.com/books/1/89/1/html/2/images/12fig01.gif

[7] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9.

[8] Java - Networking [online]. [cit. 2015-07-25]. Dostupné z: http://www.tutorialspoint.com/java/java_networking.htm

[9] SCHILDT, Herbert. Java 7: výukový kurz. 1. vyd. Brno: ComputerPress, 2012, 664 s. ISBN 978-80-251-3748-2.

[10] Java Timer and TimerTaskExampleTutorial [online]. [cit. 2015-07-26]. Dostupné z: http://www.journaldev.com/1050/java-timer-and-timertask-example-tutorial

[11] Webové aplikace - FI WIKI [online]. [cit. 2015-07-27]. Dostupné z: https://kore.fi.muni.cz/wiki/index.php/Webov%C3%A9_aplikace

[12] Java Servlet Technology [online]. [cit. 2015-07-27]. Dostupné z: http://www.oracle.com/technetwork/java/index-jsp-135475.html

[13] The Essentials of Filters [online]. [cit. 2015-07-27]. Dostupné z: http://www.oracle.com/technetwork/java/filters-137243.html

[14] JavaServer Pages Technology [online]. [cit. 2015-07-27]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[15] JSP Standard Tag Library (JSTL) Tutorial [online]. [cit. 2015-07-27]. Dostupné z: http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

[16] GlassFish - Wikipedia, the free encyclopedia [online]. [cit. 2015-07-28]. Dostupné z: https://en.wikipedia.org/wiki/GlassFish

[17] Java DB 10.11 Documentation Overview [online]. [cit. 2015-07-28]. Dostupné z: http://docs.oracle.com/javadb/support/overview.html

[18] Apache Derby - Wikipedia, the free encyclopedia [online]. [cit. 2015-07-28]. Dostupné z: https://en.wikipedia.org/wiki/Apache_Derby

[19] Java na webovém serveru: práce s databází - Zdroják [online]. [cit. 2015-07-28]. Dostupné z: http://www.zdrojak.cz/clanky/java-na-webovem-serveru-prace-s-databazi

[20] JDBC Overview [online]. [cit. 2015-07-28]. Dostupné z: http://www.oracle.com/technetwork/java/overview-141217.html

[21] Java Database Connectivity - Wikipedia, th free encyclopedia [online]. [cit. 2015-07-29]. Dostupné z: https://en.wikipedia.org/wiki/Java_Database_Connectivity

[22] Singleton Design Pattern [online]. [cit. 2015-07-29]. Dostupné z: http://www.blackwasp.co.uk/Singleton.aspx

[23] 4 - Inz2.pdf [online]. [cit. 2015-07-29]. Dostupné z: http://wiki.cs.vsb.cz/images/2/2a/Inz2.pdf

[24] datab\341ze - material-4.pdf [online]. [cit. 2015-07-29]. Dostupné z: http://vyuka.greendot.cz/materialy/material-4.pdf

[25] vazebni-tabulka.jpg [online]. [cit. 2015-07-29]. Dostupné z: http://lucie.zolta.cz/images/skola/Databaze/vazebni-tabulka.jpg

[26] Grafické uživatelské rozhraní - Wikipedie [online]. [cit. 2015-07-29]. Dostupné z: https://cs.wikipedia.org/wiki/Grafick%C3%A9_u%C5%BEivatelsk%C3%A9_rozhran%C3%AD

[27] javaduidesktopapplication001.png [online]. [cit. 2015-07-29]. Dostupné z: http://www.javaguicodexample.com/javadesktopguinetbeans4_files/javaguidesktopapplication001.png

Seznam příloh

Součástí BP je DVD.

Obsah přiloženého DVD:

1. Bakalářská práce.pdf - *vlastní práce*
2. Aplikace/Database - *databáze k aplikaci*
3. Aplikace/GameServer - *desktopová aplikace*
4. Aplikace/GameServerWebNew - *webová aplikace*
5. Programátorská dokumentace